Module:Citation/CS1/COinS

{{#lst:Module:Citation/CS1/doc|header}} This page contains various functions render a cs1|2 template's metadata.

{{#lst:Module:Citation/CS1/doc|module_components_table}}

Other documentation:

- Module talk:Citation/CS1/COinS

```
--[[--------------------------< F O R W A R D   D E C L A R A T I O N S >--------------------------------
]]

local is_set, in_array, remove_wiki_link, strip_apostrophe_markup;
-- functions in Module:Citation/CS1/Utilities

local cfg;
-- table of configuration tables that are defined in Module:Citation/CS1/Configuration


--[[--------------------------< M A K E _ C O I N S _ T I T L E >----------------------------------

Makes a title for COinS from Title and / or ScriptTitle (or any other name-script pairs)

Apostrophe markup (bold, italics) is stripped from each value so that the COinS metadata isn't corrupted with strings
of %27%27...

]]

local function make_coins_title (title, script)
	if is_set (title) then
		title = strip_apostrophe_markup (title);
-- strip any apostrophe markup
	else
		title='';
-- if not set, make sure title is an empty string
	end
	if is_set (script) then
		script = script:gsub ('^%l%l%s*:%s*', '');
-- remove language prefix if present (script value may now be empty string)
		script = strip_apostrophe_markup (script);
-- strip any apostrophe markup
	else
		script='';
-- if not set, make sure script is an empty string
	end
	if is_set (title) and is_set (script) then
```

```
                script = ' ' .. script;
-- add a space before we concatenate
        end
        return title .. script;
-- return the concatenation
end


--[[--------------------------< E S C A P E _ L U A _ M A G I C _ C H A R S
>--------------------------------

Returns a string where all of lua's magic characters have been escaped.  This
is important because functions like
string.gsub() treat their pattern and replace strings as patterns, not
literal strings.
]]

local function escape_lua_magic_chars (argument)
        argument = argument:gsub("%%", "%%%%");
-- replace % with %%
        argument = argument:gsub("([%^%$%(%)%.%[%]%*%+%-%?])", "%%%1");
-- replace all other lua magic pattern characters
        return argument;
end


--[[--------------------------< G E T _ C O I N S _ P A G E S >--------------
---------------------------------

Extract page numbers from external wikilinks in any of the |page=, |pages=,
or |at= parameters for use in COinS.

]]

local function get_coins_pages (pages)
        local pattern;
        if not is_set (pages) then return pages; end
-- if no page numbers then we're done
        while true do
                pattern = pages:match("%[(%w*:?//[^ ]+%s+)[%w%d].*%]");
-- pattern is the opening bracket, the url and following space(s): "[url "
                if nil == pattern then break; end
-- no more urls
                pattern = escape_lua_magic_chars (pattern);
-- pattern is not a literal string; escape lua's magic pattern characters
                pages = pages:gsub(pattern, "");
-- remove as many instances of pattern as possible
        end
        pages = pages:gsub("[%[%]]", "");
-- remove the brackets
        pages = pages:gsub("–", "-" );
```

```
-- replace endashes with hyphens
        pages = pages:gsub("&%w+;", "-" );
-- and replace html entities (&ndash; etc.) with hyphens; do we need to
replace numerical entities like &#32; and the like?
        return pages;
end


--[=[-------------------------< C O I N S _ R E P L A C E _ M A T H _ S T R I
P M A R K E R >-----------------

There are three options for math markup rendering that depend on the editor's
math preference settings.  These
settings are at [[Special:Preferences#mw-prefsection-rendering]] and are
        PNG images
        TeX source
        MathML with SVG or PNG fallback

All three are heavy with html and css which doesn't belong in the metadata.

Without this function, the metadata saved in the raw wikitext contained the
rendering determined by the settings
of the last editor to save the page.

This function gets the rendered form of an equation according to the editor's
preference before the page is saved.  It
then searches the rendering for the text equivalent of the rendered equation
and replaces the rendering with that so
that the page is saved without extraneous html/css markup and with a
reasonably readable text form of the equation.

When a replacement is made, this function returns true and the value with
replacement; otherwise false and the intital
value.  To replace multipe equations it is necesary to call this function
from within a loop.

]=]

local function coins_replace_math_stripmarker (value)
        local stripmarker = cfg.stripmarkers['math'];
        local rendering = value:match (stripmarker);
-- is there a math stripmarker

        if not rendering then
-- when value doesn't have a math stripmarker, abandon this test
                return false, value;
        end
        rendering = mw.text.unstripNoWiki (rendering);
-- convert stripmarker into rendered value (or nil? ''? when math render
error)
        if rendering:match ('alt="[^"]+"') then
```

```
-- if PNG math option
                rendering = rendering:match ('alt="([^"]+)"');
-- extract just the math text
        elseif rendering:match ('$%s+.+%s+%$') then
-- if TeX math option; $ is legit character that is escapes as \$
                rendering = rendering:match ('$%s+(.+)%s+%$')
-- extract just the math text
        elseif rendering:match ('<annotation[^>]+>.+</annotation>') then
-- if MathML math option
                rendering = rendering:match
('<annotation[^>]+>(.+)</annotation>')                -- extract just the
math text
        else
                return false, value;
-- had math stripmarker but not one of the three defined forms
        end
        return true, value:gsub (stripmarker, rendering, 1);
end


--[[--------------------------< C O I N S _ C L E A N U P >------------------
----------------------------------

Cleanup parameter values for the metadata by removing or replacing invisible
characters and certain html entities.

2015-12-10: there is a bug in mw.text.unstripNoWiki ().  It replaces math
stripmarkers with the appropriate content
when it shouldn't.  See https://phabricator.wikimedia.org/T121085 and
Wikipedia_talk:Lua#stripmarkers_and_mw.text.unstripNoWiki.28.29

TODO: move the replacement patterns and replacement values into a table in
/Configuration similar to the invisible
characters table?

]]

local function coins_cleanup (value)
        local replaced = true;
-- default state to get the do loop running

        while replaced do
-- loop until all math stripmarkers replaced
                replaced, value = coins_replace_math_stripmarker (value);
-- replace math stripmarker with text representation of the equation
        end

        value = value:gsub (cfg.stripmarkers['math'], "MATH RENDER ERROR");
-- one or more couldn't be replaced; insert vague error message
        value = mw.text.unstripNoWiki (value);
-- replace nowiki stripmarkers with their content
```

```
        value = value:gsub ('<span class="nowrap" style="padding%-
left:0%.1em;">&#39;(s?)</span>', "'%1");        -- replace {{'}} or {{'s}}
with simple apostrophe or apostrophe-s
        value = value:gsub (' ', ' ');
-- replace   entity with plain space
        value = value:gsub ('\226\128\138', ' ');
-- replace hair space with plain space
        if not mw.ustring.find (value, cfg.indic_script) then
-- don't remove zero width joiner characters from indic script
                value = value:gsub ('&zwj;', '');
-- remove &zwj; entities
                value = mw.ustring.gsub (value,
'[\226\128\141\226\128\139\194\173]', '');        -- remove zero-width
joiner, zero-width space, soft hyphen
        end
        value = value:gsub ('[\009\010\013]', ' ');
-- replace horizontal tab, line feed, carriage return with plain space
        return value;
end


--[[--------------------------< C O I N S >--------------------------------
----------------------------------

COinS metadata (see <http://ocoins.info/>) allows automated tools to parse
the citation information.

]]

local function COinS(data, class)
        if 'table' ~= type(data) or nil == next(data) then
                return '';
        end

        for k, v in pairs (data) do
-- spin through all of the metadata parameter values
                if 'ID_list' ~= k and 'Authors' ~= k then
-- except the ID_list and Author tables (author nowiki stripmarker done when
Author table processed)
                        data[k] = coins_cleanup (v);
                end
        end

        local ctx_ver = "Z39.88-2004";
        -- treat table strictly as an array with only set values.
        local OCinSoutput = setmetatable( {}, {
                __newindex = function(self, key, value)
                        if is_set(value) then
                                rawset( self, #self+1, table.concat{ key,
'=', mw.uri.encode( remove_wiki_link( value ) ) } );
                        end
```

```
                end
        });
        if in_array (class, {'arxiv', 'biorxiv', 'citeseerx', 'ssrn',
'journal', 'news', 'magazine'}) or (in_array (class, {'conference',
'interview', 'map', 'press release', 'web'}) and is_set(data.Periodical)) or
                ('citation' == class and is_set(data.Periodical) and not
is_set (data.Encyclopedia)) then
                        OCinSoutput.rft_val_fmt =
"info:ofi/fmt:kev:mtx:journal";                         -- journal metadata
identifier
                        if in_array (class, {'arxiv', 'biorxiv', 'citeseerx',
'ssrn'}) then        -- set genre according to the type of citation template
we are rendering
                                OCinSoutput["rft.genre"] = "preprint";
-- cite arxiv, cite biorxiv, cite citeseerx, cite ssrn
                        elseif 'conference' == class then
                                OCinSoutput["rft.genre"] = "conference";
-- cite conference (when Periodical set)
                        elseif 'web' == class then
                                OCinSoutput["rft.genre"] = "unknown";
-- cite web (when Periodical set)
                        else
                                OCinSoutput["rft.genre"] = "article";
-- journal and other 'periodical' articles
                        end
                        OCinSoutput["rft.jtitle"] = data.Periodical;
-- journal only
                        OCinSoutput["rft.atitle"] = data.Title;
-- 'periodical' article titles

-- these used only for periodicals
                        OCinSoutput["rft.ssn"] = data.Season;
-- keywords: winter, spring, summer, fall
                        OCinSoutput["rft.chron"] = data.Chron;
-- free-form date components
                        OCinSoutput["rft.volume"] = data.Volume;
-- does not apply to books
                        OCinSoutput["rft.issue"] = data.Issue;
                        OCinSoutput["rft.pages"] = data.Pages;
-- also used in book metadata

        elseif 'thesis' ~= class then
-- all others except cite thesis are treated as 'book' metadata; genre
distinguishes
                OCinSoutput.rft_val_fmt = "info:ofi/fmt:kev:mtx:book";
-- book metadata identifier
                if 'report' == class or 'techreport' == class then
-- cite report and cite techreport
                        OCinSoutput["rft.genre"] = "report";
                elseif 'conference' == class then
-- cite conference when Periodical not set
```

```
                        OCinSoutput["rft.genre"] = "conference";
                        OCinSoutput["rft.atitle"] = data.Chapter;
-- conference paper as chapter in proceedings (book)
                elseif in_array (class, {'book', 'citation', 'encyclopaedia',
'interview', 'map'}) then
                        if is_set (data.Chapter) then
                                OCinSoutput["rft.genre"] = "bookitem";
                                OCinSoutput["rft.atitle"] = data.Chapter;
-- book chapter, encyclopedia article, interview in a book, or map title
                        else
                                if 'map' == class or 'interview' == class
then
                                        OCinSoutput["rft.genre"] = 'unknown';
-- standalone map or interview
                                else
                                        OCinSoutput["rft.genre"] = 'book';
-- book and encyclopedia
                                end
                        end
                else        --{'audio-visual', 'AV-media-notes', 'DVD-notes',
'episode', 'interview', 'mailinglist', 'map', 'newsgroup', 'podcast', 'press
release', 'serial', 'sign', 'speech', 'web'}
                        OCinSoutput["rft.genre"] = "unknown";
                end
                OCinSoutput["rft.btitle"] = data.Title;
-- book only
                OCinSoutput["rft.place"] = data.PublicationPlace;
-- book only
                OCinSoutput["rft.series"] = data.Series;
-- book only
                OCinSoutput["rft.pages"] = data.Pages;
-- book, journal
                OCinSoutput["rft.edition"] = data.Edition;
-- book only
                OCinSoutput["rft.pub"] = data.PublisherName;
-- book and dissertation
        else
-- cite thesis
                OCinSoutput.rft_val_fmt =
"info:ofi/fmt:kev:mtx:dissertation";                        -- dissertation
metadata identifier
                OCinSoutput["rft.title"] = data.Title;
-- dissertation (also patent but that is not yet supported)
                OCinSoutput["rft.degree"] = data.Degree;
-- dissertation only
                OCinSoutput['rft.inst'] = data.PublisherName;
-- book and dissertation
        end
-- and now common parameters (as much as possible)
        OCinSoutput["rft.date"] = data.Date;
-- book, journal, dissertation
```

```
        for k, v in pairs( data.ID_list ) do
-- what to do about these? For now assume that they are common to all?
                if k == 'ISBN' then v = v:gsub( "[^-0-9X]", "" ); end
                local id = cfg.id_handlers[k].COinS;
                if string.sub( id or "", 1, 4 ) == 'info' then
-- for ids that are in the info:registry
                        OCinSoutput["rft_id"] = table.concat{ id, "/", v };
                elseif string.sub (id or "", 1, 3 ) == 'rft' then
-- for isbn, issn, eissn, etc that have defined COinS keywords
                        OCinSoutput[ id ] = v;
                elseif id then
-- when cfg.id_handlers[k].COinS is not nil
                        OCinSoutput["rft_id"] = table.concat{
cfg.id_handlers[k].prefix, v };        -- others; provide a url
                end
        end

        local last, first;
        for k, v in ipairs( data.Authors ) do
                last, first = coins_cleanup (v.last), coins_cleanup (v.first
or '');        -- replace any nowiki strip markers, non-printing or invisible
characers
                if k == 1 then
-- for the first author name only
                        if is_set(last)  and is_set(first) then
-- set these COinS values if |first= and |last= specify the first author name
                                OCinSoutput["rft.aulast"] = last;
-- book, journal, dissertation
                                OCinSoutput["rft.aufirst"] = first;
-- book, journal, dissertation
                        elseif is_set(last) then
                                OCinSoutput["rft.au"] = last;
-- book, journal, dissertation -- otherwise use this form for the first name
                        end
                else
-- for all other authors
                        if is_set(last) and is_set(first) then
                                OCinSoutput["rft.au"] = table.concat{ last,
", ", first };               -- book, journal, dissertation
                        elseif is_set(last) then
                                OCinSoutput["rft.au"] = last;
-- book, journal, dissertation
                        end
                end
        end

        OCinSoutput.rft_id = data.URL;
        OCinSoutput.rfr_id = table.concat{ "info:sid/", mw.site.server:match(
"[^/]*$" ), ":", data.RawPage };
        OCinSoutput = setmetatable( OCinSoutput, nil );
        -- sort with version string always first, and combine.
```

```
        --table.sort( OCinSoutput );
        table.insert( OCinSoutput, 1, "ctx_ver=" .. ctx_ver );  -- such as
"Z39.88-2004"
        return table.concat(OCinSoutput, "&");
end


--[[--------------------------< S E T _ S E L E C T E D _ M O D U L E S >----
---------------------------------

Sets local cfg table and imported functions table to same (live or sandbox)
as that used by the other modules.

]]

local function set_selected_modules (cfg_table_ptr, utilities_page_ptr)
        cfg = cfg_table_ptr;

        is_set = utilities_page_ptr.is_set;
-- import functions from selected Module:Citation/CS1/Utilities module
        in_array = utilities_page_ptr.in_array;
        remove_wiki_link = utilities_page_ptr.remove_wiki_link;
        strip_apostrophe_markup = utilities_page_ptr.strip_apostrophe_markup;
end


--[[--------------------------< E X P O R T E D   F U N C T I O N S >--------
---------------------------------
]]

return {
        make_coins_title = make_coins_title,
        get_coins_pages = get_coins_pages,
        COinS = COinS,
        set_selected_modules = set_selected_modules,
        }
```


Namespaces

- Module
- Discussion

Variants

Blue Gold Program Wiki

The wiki version of the Lessons Learnt Report of the Blue Gold program, documents the experiences of a technical assistance (TA) team working in a development project implemented by the Bangladesh Water Development Board (BWDB) and the Department of Agricultural Extension (DAE) over an eight+ year period from March 2013 to December 2021. The wiki lessons learnt report (LLR) is intended to complement the BWDB and DAE project completion reports (PCRs), with the aim of recording lessons learnt for use in the design and implementation of future interventions in the coastal zone.